VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

INFORMAČNÍ SYSTÉM FLORBALOVÉHO KLUBU

INFORMATION SYSTEM OF A FLOORBALL CLUB

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MICHAL HORKÝ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. VLADIMÍR BARTÍK, Ph.D.**
SUPERVISOR

BRNO 2018

Zadání bakalářské práce/21161/2017/xhorky23

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Horký Michal**
Obor: Informační technologie
Téma: **Informační systém florbalového klubu**
Information System of a Floorball Club
Kategorie: Informační systémy

Pokyny:

1. Seznamte se s principy tvorby webových aplikací a nastudujte potřebné technologie.
2. Analyzujte požadavky na informační systém florbalového klubu zahrnující evidenci hráčů, zápasů a turnajů. Dále budou evidovány tréninky, včetně účasti jednotlivých hráčů, i s případnými sankcemi za neúčast. Systém také umožní automaticky graficky vykreslit sestavu na základě přítomných hráčů.
3. Navrhněte informační systém splňující výše uvedené požadavky.
4. Navržený informační systém implementujte.
5. Zhodnoťte dosažené výsledky a navrhněte další možné pokračování tohoto projektu.

Literatura:

- Welling, L., Thomsonová, L.: PHP a MySQL: rozvoj webových aplikací. Vyd. 1. Praha: SoftPress, 2003, 910 s. ISBN 80-86497-60-7.
- Žára, O.: JavaScript - Programátorské techniky a webové technologie, Computer Press, 2015. ISBN: 978-80-251-4573-9

Pro udělení zápočtu za první semestr je požadováno:

- Body 1-3.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Bartík Vladimír, Ing., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato práce se zabývá vytvořením informačního systému pro podporu organizace akcí pořádaných florbalovým klubem. Jedná se o webovou aplikaci postavenou na frameworku Flask. Systém umožňuje trenérům získat přehled o účasti hráčů na jednotlivých akcích. Dále jim nabízí možnost graficky si zobrazovat sestavu pro jednotlivé turnaje. Organizátorům umožní jednodušší vedení všech potřebných statistik, které nahradí stávající zapisování do textových souborů. Tyto statistiky budou okamžitě po ukončení turnaje dostupné všem příznivcům ligy.

Abstract

This thesis deals with the creating of an information system to support organization of events organized by floorball club. It concerns a web application based on the Flask framework. The system allows the trainers to get an overview of the participation of players in the individual events. It also offers them the possibility to display graphically a line up for every single tournament. It allows easier management of all necessary statistics to the organizators which will replace current evidence to the text files. These statistics will be avaible immediately after the ending of the tournament to all the fans of the league.

Klíčová slova

Python, Flask, Bootstrap, informační systém, web, florbal

Keywords

Python, Flask, Bootstrap, information system, web, floorball

Citace

HORKÝ, Michal. *Informační systém florbalového klubu.* Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vladimír Bartík, Ph.D.

Informační systém florbalového klubu

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Bartíka, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michal Horký
14. května 2018

Poděkování

Rád bych poděkoval panu Ing. Vladimíru Bartíkovi, PhD. za cenné rady a vedení mé bakalářské práce. Dále bych rád poděkoval florbalovému oddílu Sokola Třebíč za poskytnutí informací potřebných pro vytvoření systému.

Obsah

Kapitola 1

Úvod

Hlavním cílem bakalářské práce je vytvoření informačního systému pro podporu organizací akcí pořádaných Tělocvičnou jednotou Sokol Třebíč. Ten umožní hráčům získat přehled o konaných tréninicích a turnajích. Trenérům zjednoduší organizaci tréninků, včetně získání přehledu o účasti jednotlivých hráčů na tréninicích a turnajích. Zapisovatelům zjednoduší vedení statistik o zápasu a organizátorům ligy zrychlí vytváření tabulek jednotlivých statistik. Ty budou navíc generovány automaticky, takže se sníží riziko zavedení chyby při jejich vypočítávání. V neposlední řadě umožní všem příznivcům ligy okamžitý přístup ke všem statistikám.

Aktuálně používaný způsob vedení statistik neumožňuje evidovat všechny důležité informace o akcích, které klub pořádá. Zároveň je aktuální způsob zaznamenávání a počítání statistik zdlouhavý, což neumožňuje mít aktuální přehled o stavu ligy. Další velkou nevýhodou je náchylnost na zavedení chyb do statistik a časová náročnost na jejich odhalení a nápravu. Z tohoto důvodu bylo nutné vytvořit informační systém, který veškerou práci zjednoduší a urychlí.

Aby bylo možné používat systém na široké škále zařízení, je realizován jako webová aplikace. Díky tomu je možné zobrazovat výsledky i na mobilních zařízeních. Zároveň je díky tomu umožněn okamžitý přístup ke všem statistikám. Tyto statistiky je možné zobrazovat i v průběhu turnaje, stejně jako je možné si zobrazovat průběžné výsledky zápasů konaných na turnaji. Toho můžou využít i hráči, kteří tak jednoduše zjistí, jak si na turnaji vedou napříč všemi statistikami.

Pro pohodlné používání systému na mobilních zařízeních je nutné, aby byl web responzivní. To je zajištěno s pomocí knihovny Bootstrap. Díky jejímu použití je vzhled systému optimalizován pro různé velikosti zobrazovacích zařízení. Nejedná se tedy pouze o optimalizaci pro mobilní zařízení, ale i pro jiná zařízení, kde se může lišit velikost obrazovky.

Obsah práce je rozdělen do několika kapitol, které postupně popisují fáze vývoje informačního systému. První kapitolu představuje tento úvod. Druhá kapitola se zabývá principem tvorby webových aplikací a seznamuje čtenáře s dostupnými programovacími jazyky. Třetí kapitola uvádí podrobnější informace o technologiích, které jsou pro tvorbu informačního systému použity. Následující kapitola popisuje požadavky na vyvíjený systém. To zahrnuje konkrétní požadavky na funkčnost, a také požadavky na uživatelské role. Další kapitola má za úkol seznámit čtenáře s návrhem informačního systému, který byl vytvořen na základě analýzy požadavků. Následující dvě kapitoly potom popisují samotnou implementaci návrhu systému a testování této implementace. Poslední kapitolou je závěr, který obsahuje také návrhy na další možnosti vývoje a rozšíření.

Kapitola 2

Principy tvorby webových aplikací

Tato kapitola popisuje, co to webová aplikace je a její důležité části. Zabývá se principy tvorby webových aplikací a technologiemi, které se k tvorbě používají. Vybrané technologie budou popsány podrobněji v kapitole 3.

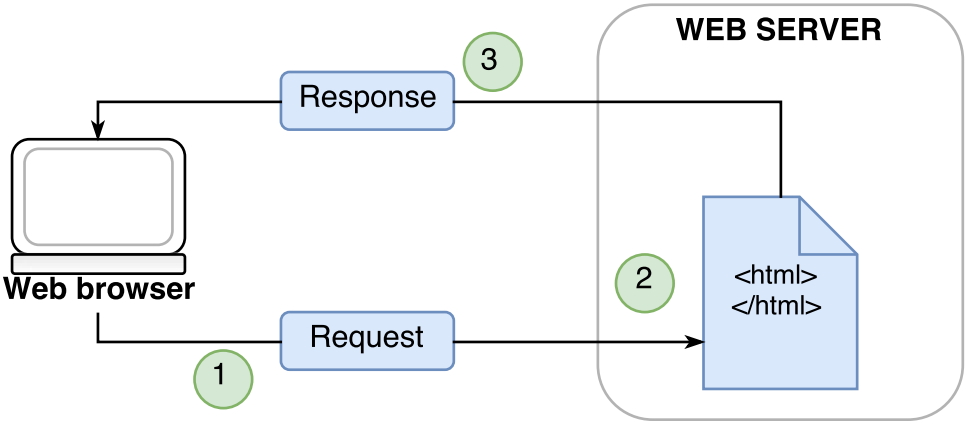
2.1 Webová aplikace

Webová aplikace je webové místo, které obsahuje stránky s určitým obsahem. Tento obsah může být buď předem určený nebo neurčený. Konečný obsah stránky s neurčeným obsahem se vytvoří až tehdy, kdy klient požádá o stránku z webového serveru. Konečný obsah takové stránky závisí na akcích návštěvníka, a protože se může měnit, říká se takové stránce dynamická. Hlavním cílem webových aplikací je řešit nejrůznější úkoly a problémy. [3]

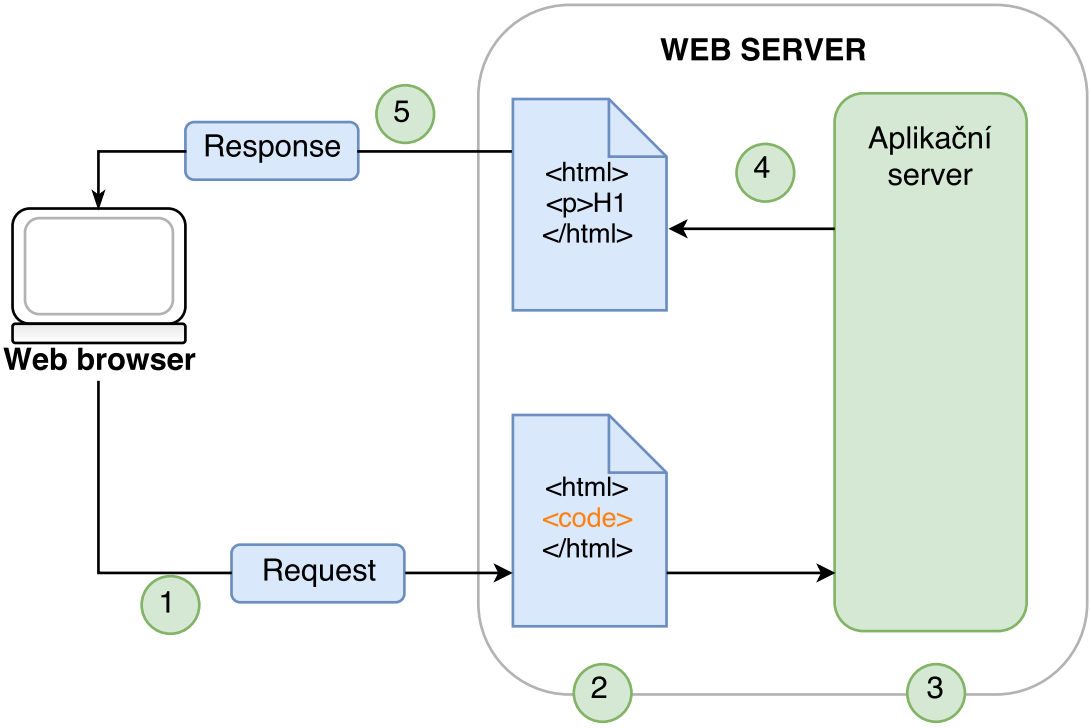
Webovou aplikaci tvoří kolekce statických a dynamických webových stránek. Statická stránka má neměnný obsah. Pokud o ni návštěvník požádá, webový server mu ji zašle beze změny. U dynamické stránky se tento proces liší. Pokud o dynamickou stránku návštěvník požádá, je tato stránka nejprve na webovém serveru modifikována, a teprve potom je odeslána návštěvníkovi. Právě kvůli proměnlivé povaze obsahu stránky se jí říká dynamická. [3]

Statické webové místo je tvořeno množinou příslušných HTML stránek a souborů, které jsou umístěny v počítači, na kterém je webový server spuštěn. Webový server je software, který obsluhuje jednotlivé požadavky, které jsou mu odesílány z webových prohlížečů návštěvníků. Tento požadavek je vytvořen ve chvíli, kdy návštěvník klikne na nějaký odkaz, vybere záložku nebo například zadá přímo URL adresu požadované stránky. Obsah statické webové stránky je určován návrhářem a nemění se v čase. Vždy, když návštěvník o tuto stránku požádá, je mu odeslána ve stejné podobě, jako byla navržena návrhářem. [3] Tento princip je zobrazen na obrázku 2.1.

Pokud webový server obdrží požadavek na dynamické webové stránky, je při zpracování požadavku uplatňován rozdílný princip než u statických. Jak již bylo zmíněno, statická stránka je odeslána návštěvníkovi beze změny. U dynamických stránek je po přijetí požadavku stránka nejprve předána aplikačnímu serveru. Ten podle instrukcí v kódu stránku dokončí a následně tento kód ze stránky odstraní. Výsledkem tohoto procesu je statická stránka, která je aplikačním serverem předána zpět webovému serveru a ten ji odesílá návštěvníkovi, který o ni požádal. Stránka, kterou prohlížeč po tomto zpracování obdrží, již obsahuje pouze čisté HTML. [3] Tento princip je zobrazen na obrázku 2.2.



Obrázek 2.1: Zpracování statické webové stránky [3]



Obrázek 2.2: Zpracování dynamických stránek [3]

2.2 HTML, CSS

HTML (*Hyper Text Markup Language*) je značkovací jazyk používaný pro vytváření webových stránek. Popisuje strukturu webu pomocí značek. Stavebními kameny stránek jsou HTML elementy. Tyto elementy jsou reprezentovány tagy. Tag je název elementu, který je obalen znaky většítka a menšítka. Prohlížeče tagy nezobrazují, ale využívají je k vykreslení stránky. [1]

CSS (*Cascading Style Sheets*) popisuje, jakým způsobem mají být jednotlivé HTML elementy zobrazeny. Mohou ušetřit mnoho práce, protože je díky nim možné starat se o vzhled více webových stránek naráz. CSS může být do HTML přidáno třemi způsoby:

- Inline – použitím atributu `style` v HTML elementu
- Internal – použitím elementu `<style>` v sekci `<head>`
- External – použitím externího CSS souboru

Nejčastější způsob přidávání CSS je udržování stylů v odděleném CSS souboru. [2]

2.3 Programovací jazyky

Pro vývoj webových aplikací je možné využít různé programovací jazyky a frameworky. Nejvíce používané je PHP, ale existuje několik alternativ. Níže jsou uvedeny vybrané jazyky a frameworky.

2.3.1 PHP

PHP (*Hypertext Preprocessor*) je široce používaný skriptovací jazyk, který je velice vhodný pro webový vývoj a lze jej vložit do HTML. Lze ho použít také pro tvorbu desktopových aplikací a programování v příkazovém řádku. PHP stránky obsahují HTML s vloženým kódem, který se provádí. Kód je uzavřen do speciálních instrukcí, které umožňují oddělit kód od zbytku dokumentu. Kód je spouštěn na serveru, kde se vygeneruje HTML dokument a ten je následně odeslán klientovi. Klient obdrží výsledek spuštěného skriptu, ale nezjistí jaký byl základní kód. [6]

Mezi nejpoužívanější frameworky patří:

- Laravel,
- Code Igniter,
- Symfony.

2.3.2 Python

Python je moderní programovací jazyk vyvinutý Guido van Rossumem. Jedná se o platformě nezávislý jazyk a lze jej používat pro psaní malých skriptů, aplikací nebo vývoj velkých softwarových projektů. Poskytuje přístup k velmi výkonnému uživatelskému rozhraní. Zvláštností jeho syntaxe je, že bloky kódu nejsou uzavřeny ve složených závorkách. Pro oddělení bloků se využívá odsazení textu. Blok se skládá z jednoho nebo více příkazů oddělených novým řádkem, kde všechny příkazy musí být na stejné úrovni odsazení. [11]

Pro webový vývoj jsou dostupné například tyto frameworky:

- Django,
- TurboGears,
- Flask.

2.3.3 Ruby

Ruby je interpretovaný jazyk. Může být nazýván skriptovacím nebo objektově orientovaným. Ruby překračuje pouhé skriptování, i když programy můžou vypadat jako shell skripty. Není to pouze procedurální jazyk, ale je možné ho tak využívat. Ruby má jednoduchou syntaxi, která je velice snadná na naučení. Pro webový vývoj existuje framework, který se jmenuje Ruby on Rails. Tento framework je vhodný pro vývoj databázově založených webových stránek. [10]

2.3.4 C#

C# je elegantní a typově bezpečný objektově orientovaný jazyk, který umožňuje vývojářům vytvářet různé bezpečné a robustní aplikace, které běží na .NET frameworku. Syntaxe je založena na složených závorkách, podobně jako tomu je například u C, C++ nebo Javy. Díky tomu jsou vývojáři, kteří některý z těchto jazyků znají, schopni velice rychle pracovat i v jazyce C#. Jako objektově orientovaný jazyk podporuje koncepty zapouzdření, dědičnosti a polymorfismu. Pro webový vývoj je dostupný framework ASP.NET. [15]

2.3.5 JavaScript

Jazyk JavaScript je navržen pro skriptování v prostředí webového prohlížeče. Jeho syntaxe vychází z jazyka C, některá standardní rozhraní se podobají Javě, objektový a funkcionální model je inspirovaný jazyky Scheme a Self. Je dynamicky typovaný a funkce jsou v něm k dispozici jako běžný datový typ. Pro serverové vykonávání JavaScriptu se využívá projekt Node.js. Node.js je sada rozhraní a knihoven, obsahující implementaci V8. Díky tomu umožňuje jednoduše spouštět JavaScriptový kód i mimo webový prohlížeč. [17]

Kapitola 3

Použité technologie

Pro tvorbu informačního systému je nutné si stanovit, v čem se budou jednotlivé části programovat a jaké technologie k tomu budou využity. Následující část se zabývá popisem vybraných technologií a je zde uveden stručný důvod tohoto výběru.

3.1 Server (back-end)

Pro programování serveru bylo prvním kandidátem PHP. Jak již bylo zmíněno, je pro webový vývoj nejpoužívanějším jazykem, a díky tomu je zde velká podpora komunity a velké množství různé literatury. Vzhledem k některým specifikům syntaxe tohoto jazyka, které autorovi úplně nevyhovovaly, pokračoval výběr dále. Další možností byl Python, který se jevil jako zajímavá varianta. Python má pro webový vývoj dva druhy frameworků. Jeden z nich je komplexní a je u něj předem dané, jaké technologie budou pro jednotlivé části systému použity, to je tzv. *full-stack* framework. Druhou možností jsou *microframeworky*. U těch samotný framework obsahuje pouze omezenou funkčnost a další technologie si může programátor zvolit sám (pro databázi, autentizaci uživatelů apod.), což mu umožňuje zvolit si technologie, které mu vyhovují. V úvahu ještě přicházely další jazyky, ale žádný již nezaujal tolik jako Python. Zbývalo tedy vybrat konkrétní framework. Nejvýraznějšími kandidáty byly Django a Flask. Django je komplexní framework, zatímco Flask je zástupce *microframeworků*. Zajímavěji se jevila varianta, kdy jednotlivé technologie nebudou předem stanoveny, a proto je ve vytvářeném systému použit Flask.

3.1.1 Flask

Tento framework je často označován jako mikro. To znamená, že se snaží udržet jednoduché, ale rozšiřitelné jádro. Flask za programátora nedělá moc rozhodnutí, jako například jakou databázi použít. V základu neobsahuje databázovou vrstvu, validaci formulářů nebo cokoliv jiného, co již nabízí jiné existující knihovny. Místo toho Flask podporuje rozšíření pro přidání funkcionality, jako kdyby byla implementována přímo ve Flasku. Rozšíření poskytují integraci databáze, validaci formulářů, správu nahrávání, různé možnosti autentizace a mnoho dalších. [14]

3.1.2 Šablonovací systém Jinja2

Šablonovací systémy umožňují vývojáři generovat požadovaný typ obsahu, jako například HTML, s použitím některých datových a programovacích konstrukcí. Pro manipulaci s vý-

stupem je tedy možné použít podmínky a cykly. Šablony jsou vytvořeny vývojářem a následně zpracovány šablonovacím systémem. Při tomto procesu jsou přepsány značky a bloky, které jsou nahrazeny daty. [4]

Flask využívá šablonovací systém Jinja2. Jinja2 je moderní šablonovací jazyk pro Python, který je modelovaný podle šablon Django. Je rychlý, široce používaný a bezpečný. [5] Pro přístup k proměnným se používají dvojité složené závorky. A pro podmínky a cykly se používá složená závorka a znak procenta.

3.2 Databáze

U databáze byl výběr jednodušší, než v případě serveru. Již od začátku bylo jasné, že se bude jednat o SQL databázi. Dalším důležitým kritériem bylo to, aby byla zdarma. Nejjednodušší variantou tedy bylo SQLite. Takováto databáze se hodí pro vývoj, protože je uložena ve formě jediného souboru na disku a je možné využít grafické rozhraní pro snadné změny a zobrazení dat v databázi. SQLite ale není vhodné pro použití ve finální verzi informačního systému, například kvůli tomu, že neumožňuje současný zápis. To by způsobovalo problémy při připojení více uživatelů.

Dalšími zástupci jsou MySQL a PostgreSQL. Hlavními rozdíly mezi nimi jsou počty funkcí a rychlost. Zatímco PostgreSQL má oproti MySQL větší počet pokročilých funkcí, tak MySQL vyniká v rychlosti. Proto bude MySQL použito při nasazení systému a pro vývoj a testování SQLite. Pro práci s databází je použita knihovna SQLAlchemy.

3.2.1 SQLAlchemy

SQLAlchemy je knihovna pro Python poskytující vysokoúrovňové rozhraní pro relační databáze jako Oracle, DB2, MySQL, PostgreSQL a SQLite. SQLAlchemy poskytuje jazyk SQL, který je nezávislý na databázovém serveru, a ORM (object-relational mapper). [8]

Díky ORM není nutné používat SQL příkazy. Vše lze napsat přímo v Pythonu a ten je následně překládán na jazyk SQL. Tabulky databáze jsou vytvořeny na základě tříd, kde jednotlivé proměnné tvoří sloupce tabulky. Všechny proměnné využívají třídu `Column`, kterou SQLAlchemy poskytuje. Knihovna také poskytuje různá nastavení pro jednotlivé sloupce tabulky. Lze tedy jednoduše určit, který sloupec bude primárním klíčem, cizím klíčem nebo bude obsahovat unikátní hodnotu.

Vztahy mezi tabulkami jsou řešeny pomocí cizích klíčů. Navíc je tu ale možnost v jednotlivých třídách vytvořit další proměnné, které budou obsahovat objekty, popřípadě seznamy objektů, které jsou s příslušnou tabulkou spojeny pomocí cizího klíče. Díky tomu je možné k připojené tabulce přistupovat, aniž by bylo nutné explicitně vyžadovat tuto tabulku již při dotazu. Všechny tyto proměnné jsou vytvářeny s pomocí metody `relationship`, kterou poskytuje SQLAlchemy.

3.3 Klient (front-end)

Pro vytváření vzhledu stránek, které se budou zobrazovat na straně klienta, je využito technologií HTML a CSS. Dále je využito šablonovacího systému Jinja2, díky kterému je možné do HTML vložit bloky kódu, které se zpracují na serveru. Výsledný HTML dokument potom bude obsahovat pouze data, která vznikla zpracováním kódu na serveru. Pro definici vzhledu není vytvořeno vlastní CSS, ale je využita knihovna, které nabízí zdarma vytvořené CSS soubory. To umožňuje používat dobře vypadající vzhled i vývojářům, kteří nejsou

graficky velmi zdatní. Takto získaný vzhled je možné i upravovat. Tato knihovna se nazývá Bootstrap. Pro grafické vykreslování sestavy a práci s formuláři je využito JavaScriptu.

3.3.1 Bootstrap

Díky knihovně Bootstrap[1] může vzhled stránek vytvářet každý programátor, který nemá rád tvorbu vzhledu nebo by nedokázal vytvořit vzhled, který by vypadal profesionálně. Stačí si zvolit jednu z mnoha dostupných šablon a stáhnout si příslušné soubory. Poté stačí jen v HTML využívat třídy definované ve staženém CSS. Bootstrap využívá responzivního designu, což je velice výhodné, pokud budou vytvářené stránky zobrazovány na zařízeních s různou velikostí obrazovky. Další výhodou je využití JavaScriptu, čímž je umožněno používání dynamických prvků, jako například výsuvného menu a podobně.

3.4 Architektura MVC

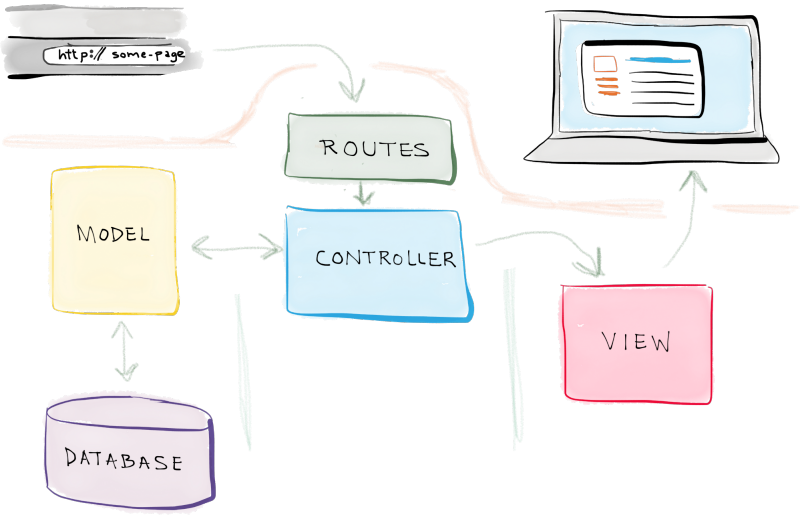
MVC je architektonický vzor, který se uchytil zejména pro vývoj webových aplikací. Jeho základní myšlenkou je oddělení logiky od výstupu. Snaží se tedy odstranit situaci, kdy je vše v jednom souboru (databázové dotazy, různé operace a HTML tagy). K tomu využívá tři komponenty, které se nazývají Model, View (pohled) a Controller (kontroler). [16] Princip MVC je zobrazen na obrázku 3.1.

Model zajišťuje získání dat. Neví nic o výstupu, jeho úloha spočívá v přijetí parametrů a vydání příslušných dat. Pro model je nepodstatné, odkud požadavek přišel a jakým způsobem budou výstupní data zformátována a zobrazena. [16] V našem případě funkci modelu zajišťuje SQLAlchemy, které pomáhá v komunikaci s databází.

Pohled se stará o zobrazení výstupu uživateli. Nejčastěji se jedná o HTML stránku, která může obsahovat kód, jenž do dokumentu umožňuje vkládat proměnné, popřípadě umožňuje vkládat cykly (v našem případě to umožňuje Jinja2). Samotné pohledy je možné vkládat do sebe. Je tedy například možné mít samostatně uložené menu, které se vkládá do ostatních stránek, díky čemuž není nutné měnit menu v každém souboru zvlášť. [16]

Kontroler zajistí funkčnost celého MVC. Zastává funkci prostředníka, který komunikuje s uživatelem, modelem i pohledem. Kontroler dle parametrů v URL zjistí, co má udělat. Od modelu získá potřebná data, která předá pohledu a ten se následně zobrazí. [16]

[1] Dostupné na: https://getbootstrap.com/



Obrázek 3.1: Architektura MVC [13]

3.5 AJAX

AJAX (*Asynchronous JavaScript and XML*) je soubor technik pro vytváření vysoce interaktivních webových stránek a webových aplikací. V těchto aplikacích je kladen důraz na aktualizaci webové stránky s pomocí dat získaných ze serveru bez nutnosti obnovovat v prohlížeči celou stránku. Je tedy možné zaslat požadavek na server, získat data a ty zobrazit na aktuální stránce, aniž by se uživateli musela obnovovat stránka. To by totiž mohlo na uživatele při častém opakování působit rušivě. Příkladem může být například zobrazení nápovědy při použití vyhledávačů. [12] V našem případě je AJAX využit zejména pro doplňování formulářů, kdy některé části závisí na předchozím výběru uživatele.

Kapitola 4

Analýza požadavků

V následující kapitole se budeme věnovat současnému řešení a požadavkům na informační systém. Cílem informačního systému je především zjednodušení organizace turnajů. Trenérům systém umožní získat přehled o účasti na trénincích a turnajích. Hráči budou mít možnost jednoduše zjistit všechny nadcházející termíny a zobrazit si své aktuální statistiky. Všechny výsledky zápasů a turnajů budou také okamžitě k dispozici všem příznivcům florbalové ligy.

Všechny požadavky je nutné zpracovat a ujasnit si detaily. Protože se často stává, že se nějaká informace opomene, bylo zapotřebí se scházet a požadavky doplňovat. Na těchto schůzkách se vždy probral aktuální stav, vyřešily se části, na které se zapomnělo nebo nebyly úplně odladěny a dohodl se další postup. Schůzky probíhaly průběžně dle aktuálně implementovaných součástí systému.

4.1 Nedostatky současného řešení

Momentálně je celá správa ligy řešena pouze pomocí papírových zápisů o utkáních a turnajích. Všechny statistiky jsou dopočítávány ručně, díky čemuž vzniká velké riziko chyby. Zveřejnění statistik trvá dlouho, takže se týmy i hráči dozvídají aktuální stav ligy až se značným zpožděním.

Vzhledem k velkému počtu hráčů není možné tímto způsobem řešit organizace tréninků. Trenéři ve většině případů neví, kolik hráčů se dostaví na trénink a nemají ani přehled o celkové účasti jednotlivých hráčů. Obdobný problém vzniká i na turnajích.

4.2 Sezóny a kategorie

Základem celého systému jsou sezóny a kategorie, od kterých se odvíjí všechna ostatní data. Sezóny je nutné ukládat z hlediska udržování historie, a také kvůli možnosti změny kategorií mezi jednotlivými ročníky. U kategorií se mezi jednotlivými ročníky mění i maximální věk hráčů, kteří mohou v dané kategorii nastoupit. V rámci kategorií se dále může mezi jednotlivými ročníky měnit i délka zápasu. Ta závisí vždy na věku hráčů a na počtu týmů, které se účastní ligy v dané sezóně a kategorii.

4.3 Hráči a týmy

Pro organizaci ligy a tréninků je nezbytné vést informace o všech týmech a hráčích. Každý tým musí spadat do některé z kategorií a všichni hráči, kteří budou za tento tým hrát, musí splňovat věkové omezení příslušné kategorie. Počet hráčů, kteří do jednotlivých týmů spadají, je omezený. Konkrétně může být v každém týmu maximálně 20 hráčů a systém musí toto omezení automaticky kontrolovat. V systému je nutné ukládat informaci o tom, zda se jedná o vlastní tým, nebo je to tým jiného klubu. Pokročilé funkce systému (které jsou popsány níže) budou potom dostupné pouze vlastním týmům a hráčům v nich. U hráčů, kteří jsou členové vlastního týmu, bude oproti ostatním hráčům nutné ukládat ještě další informace. Konkrétně se bude jednat o standardní formaci a pozici hráče.

4.4 Tréninky

Systém bude umožňovat vytváření tréninků pro jednotlivé kategorie. Tréninky budou vždy probíhat každý týden. Protože se v průběhu sezóny koná velké množství tréninků pro různé kategorie, bude při vytváření tréninků zadáno pouze datum prvního a posledního tréninku. Všechny tréninky mezi těmito dvěma daty se potom vytvoří automaticky. U každého tréninku bude možné zadat místo, čas začátku a délku konání. Jednotlivé tréninky bude možné odstranit, může se totiž stát, že bude některý trénink z různých důvodů zrušen.

Ze všech tréninků budou mít hráči vlastních týmu možnost se omlouvat. Díky tomu si trenéři budou moci zobrazit předpokládanou účast na jednotlivé tréninky. Dále musí mít trenéři možnost po každém tréninku zadat, kteří hráči se daného tréninku zúčastnili. Díky tomu se získá celkový přehled o účasti hráčů na trénincích. Na základě těchto statistik bude možné automaticky počítat pokuty za neomluvené tréninky.

4.5 Turnaje a zápasy

V systému bude možné vytvářet turnaje a zápasy. Při vytváření turnaje musí být možné zadat datum, čas začátku, místo konání a pořadí turnaje. Pro turnaj musí být možné vytvořit soupisku z hráčů, kteří patří do týmů, které se turnaje účastní. Na zápisech o utkání potom budou uvedeni pouze tito hráči, jiní se již zápasů nesmí zúčastnit.

Na turnaje musí mít hráči vlastních týmů možnost se omlouvat. Na základě této informace a standardní pozici a formaci hráče si trenéři budou moci graficky zobrazit předpokládanou sestavu pro daný turnaj. Druhou možností bude zobrazení sestavy na základě soupisky turnaje. Může se totiž stát, že některý z hráčů, který se neomluvil, nakonec na turnaj nepřijde.

Podobně jako u tréninků bude možno automaticky počítat pokuty za neomluvené turnaje. Místo zadávání účasti se zde použije soupiska turnaje. Pokuty za tréninky a turnaje budou počítány dohromady.

U zápasu musí být možné zadat, kdo vedl zápas jako rozhodčí, kdo obsluhoval časomíru, a kdo zapisoval statistiky. Aby nebylo nutné jména těchto osob neustále zapisovat, bude možné je v systému vytvářet a k zápasům pouze přiřazovat. Dále se pro jednotlivé hráče budou ukládat osobní statistiky. Konkrétně se bude jednat o góly, asistence, trestné minuty, odchytané minuty a obdržené branky. Ve většině případů bude možné některé z těchto statistik zanedbat. Hráč totiž bude buď brankář nebo hráč v poli. Ve výjimečných případech

může ale během zápasu nastat situace, kdy se brankář zraní a jeden z hráčů bude po zbytek zápasu brankářem. V takovém případě bude nutné ukládat všechny výše zmíněné statistiky.

Pokud má tým více brankářů, je možné aby se během zápasu střídali. V takovém případě musí být umožněno zadat čas, který jeden z brankářů strávil na hřišti. U dalšího brankáře bude čas vypočítán automaticky na základě délky zápasu pro příslušnou kategorii a času zadaného u předchozího brankáře.

Dále systém musí umožňovat řešit některé specifické situace. První z nich je vlastní gól, v takovém případě se gól nezapisuje žádnému hráči, ale musí se připočítat do gólů příslušného týmu. Druhou situací je hra bez brankáře a případný gól v této situaci. Pokud místo brankáře nastoupí hráč, nesmí se brankáři počítat čas, který stráví na střídačce, do odchytaných minut. Zároveň pokud tým, který hraje bez brankáře obdrží gól, nesmí se tento gól počítat příslušnému brankáři do statistik. Samozřejmě je ale nutné připsat tento gól druhému týmu.

U zápasu musí být možnost přidat komentář. Tento komentář bude využíván především k popisu důvodu vyloučení, informaci o oddechovém čase apod. Kompletní zápisy jednotlivých turnajů musí být možné zobrazit. Poté je bude možné tisknout kvůli podpisům kapitánů zúčastněných týmů.

4.6 Statistiky

Systém musí umožňovat generování různých statistik týkajících se tréninků, zápasů a turnajů. Bude možné zobrazovat pořadí týmů, střelců, brankářů a hráčů podle kanadského bodování[1]. Bez pořadí pak budou uváděny statistiky trestných minut a účasti na trénincích a turnajích.

Všechny tyto statistiky bude možné zobrazovat v rámci jednotlivých turnajů a celkově za všechny turnaje v uživatelem požadovaném ročníku a kategorii. Statistiku účasti na turnajích a trénincích bude možné zobrazovat pro všechny vlastní týmy dohromady, a také pro jednotlivé týmy zvlášť.

Základní statistikou bude tabulka pořadí týmů. V této tabulce budou o každém týmu uvedeny tyto informace:

- pořadí týmu,
- název týmu,
- počet odehraných zápasů,
- počet vítězství,
- počet remíz,
- počet porážek,
- skóre (poměr vstřelených a obdržených branek),
- počet bodů.

O pořadí týmů budou rozhodovat tyto faktory:

1. počet bodů,

[1] Za každý vstřelený gól a asistenci získává hráč jeden bod.

2. vzájemné zápasy,
3. skóre ze vzájemných zápasů,
4. vstřelené góly ve vzájemných zápasech.

Za vítězství získá tým 3 body, za remízu 1 bod a za prohru 0 bodů. V případě, že všechny tyto statistiky budou mít týmy totožné, budou se příslušné týmy dělit o jedno místo.

U každé tabulky s hráčskými statistikami bude uvedeno pořadí hráče, jméno hráče a tým, za který hráč hraje. Další specifické informace uváděné v tabulkách budou popsány níže.

V tabulce střelců bude uveden počet vstřelených gólů. Pokud bude mít více hráčů stejný počet vstřelených branek, budou tito hráči uvedeni společně na jednom místě a následně bude příslušný počet dalších míst vynechán. Pokud tedy např. na prvním místě budou 3 hráči, tak další hráč bude na 4. místě. V tabulce nebudou uváděni hráči, kteří nevstřelili žádný gól. Tabulka by totiž potom mohla obsahovat zbytečně velký počet řádků.

V tabulce kanadského bodování bude uveden počet bodů a rozložení bodů mezi góly a asistence. Při shodném počtu bodů rozhoduje počet vstřelených branek. Pokud bude mít více hráčů stejný počet bodů a zároveň stejný počet vstřelených branek, budou v tabulce uvedeni stejným způsobem jako v tabulce střelců. V kanadském bodování můžou být uvedeni i brankáři, kteří si mohou připsat asistence. Podobně jako u tabulky střelců, nebudou ani v této tabulce uváděni hráči, kteří nezískali žádný bod.

U brankářů je rozhodující počet obdržených gólů na minutu. V tabulce tedy musí být uveden počet odchytaných minut, počet obdržených gólů a počet obdržených gólů na minutu. Počet obdržených gólů na minutu se uvádí s přesností na 3 desetinná čísla. U brankářů je podmínkou zařazení do pořadí odchytání alespoň 40% času. Brankáři, kteří odchytali méně času, budou v tabulce uvedeni na posledních místech bez udaného pořadí.

Další statistikou budou trestné minuty. Hráče seřazené podle trestných minut bude možné zobrazovat v tabulce. Celkový počet trestných minut týmu bude možné zobrazovat v grafu, kde bude patrné, který tým strávil nejvíce času na trestné lavici. Ani v této tabulce nebudou uváděni hráči, kteří mají nulový počet trestných minut.

Poslední statistikou bude účast hráčů na trénincích a turnajích. Bude z ní patrné, kolik událostí proběhlo celkově, kolika se hráč zúčastnil, na kolik byl omluven a kolika se bez omluvy nezúčastnil. Tato statistika se povede pouze pro hráče vlastních týmů a bude určena především trenérům a samotným hráčům.

4.7 Uživatelé

Uživatelské účty budou vytvářeny pouze pro určité skupiny uživatelů. Pro uživatele, kteří si chtějí pouze zobrazit statistiky ligy, není nutné účty vytvářet. Skupiny, pro které je nutné vytvářet uživatelské účty, jsou:

- administrátor,
- organizátor,
- zapisovatel,
- trenér,
- hráč.

Všechny typy účtů kromě hráčských se budou v systému vytvářet ručně. Hráčské účty musí být vzhledem k velkému počtu hráčů možné vytvářet automaticky. Budou vytvářeny pouze pro hráče ve vlastních týmech a budou se skládat z příjmení, prvních dvou písmen jména a dvoumístného čísla. Zapisovatelské účty nebudou vytvářeny pro všechny zapisovatele. Ti se během turnajů často střídají mezi zápasy, protože se většinou jedná o hráče některého z týmů. Z toho důvodu by bylo nepraktické, aby se při každé takovéto změně museli odhlašovat a znovu přihlašovat do systému. Řešení spočívá v tom, že bude vytvořeno pouze omezené množství těchto účtů (konkrétní počet již bude záležet na domluvě organizátorů a administrátorů), které budou používat všichni zapisovatelé.

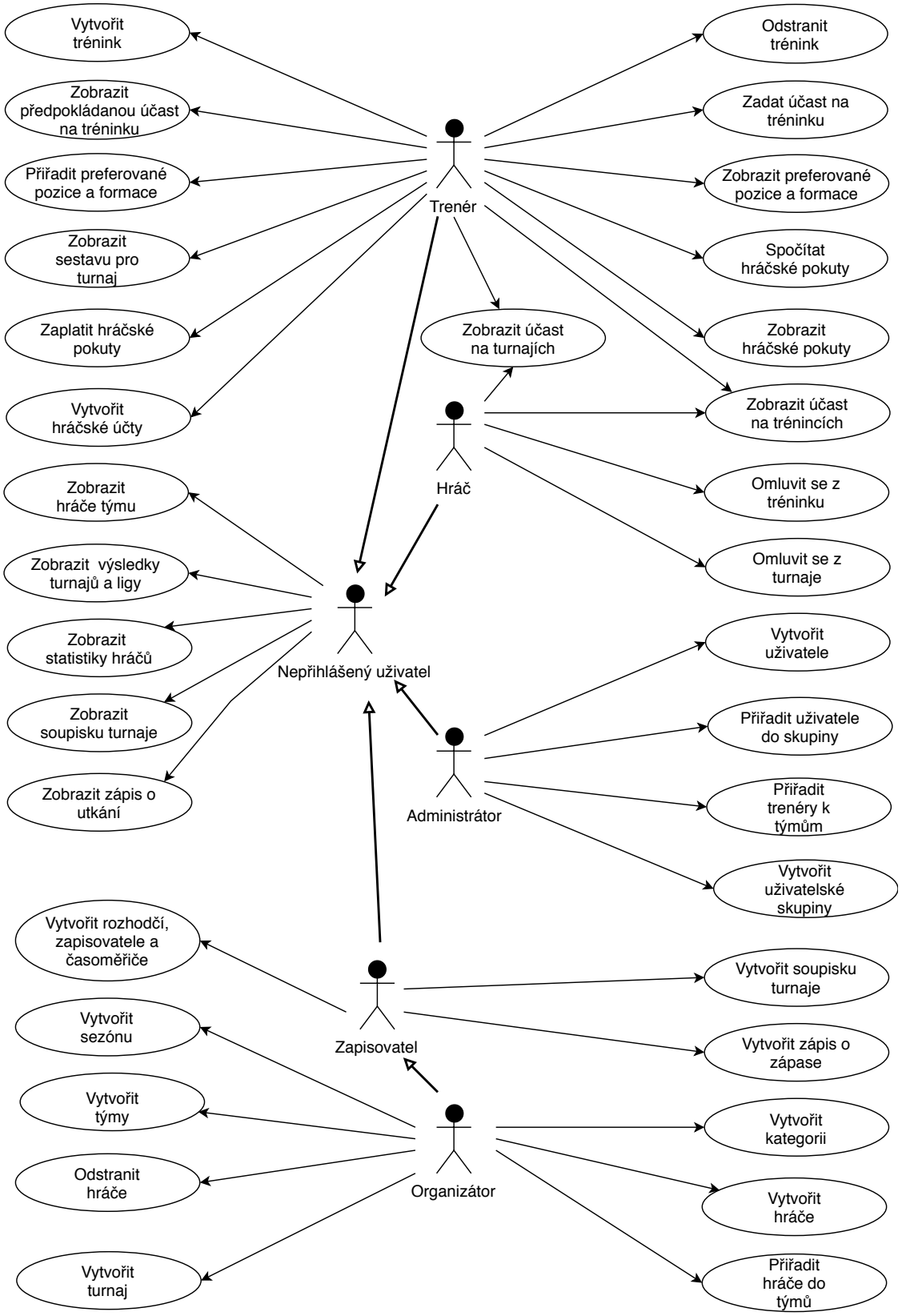
4.8 Diagram případů užití

Diagram případů užití (Use Case diagram) patří do UML (Unified Modeling Language). Nejtypičtějším účelem diagramu případů užití je dokumentování funkčních požadavků navrhovaného systému. Diagram případů užití ukazuje hranice systému stejně tak jako zodpovědnosti, a proto je dobrým nástrojem pro dokumentování rozsahu systému. [9]

Existují tři typy diagramu případů užití, které mají různá zaměření. Nejpoužívanější z nich se zaměřuje na systém. V tomto typu se účastník snaží dosáhnout cíle s využitím daného systému. Účelem tohoto typu diagramu je zachytit funkční požadavky systému. [9]

Diagram se skládá z několika částí. Nejdůležitějšími jsou aktér a případ užití. Aktér reprezentuje roli osoby, externího systému nebo zařízení, které pracuje se systémem. Případ užití reprezentuje sérií kroků (specifikované v příslušném detailu případu užití), které se dějí mezi příslušným aktérem a systémem pro získání hodnoty pro primárního aktéra. Vztah mezi aktérem a případem užití (znázorněný čárou mezi nimi) indikuje, že se aktér daného případu užití účastní. Pokud je mezi dvěma aktéry šipka, znamená to, že jeden z nich je obecný a druhý specializovaný. Obecný jen ten, ke kterému šipka směřuje. Obecný aktér se účastní všech případů užití, se kterými je ve vztahu. Specializovaný aktér se účastní všech případů užití, se kterými je ve vztahu obecný aktér a navíc všech, se kterými je ve vztahu on sám. [9]

Na obrázku 4.1 jsou zobrazeny jednotlivé akce systému a účastníci, kteří se budou těchto akcí účastnit. Zároveň obrázek dává přehled o tom, které části systému budou jednotlivým uživatelům dostupné.



Obrázek 4.1: Diagram případů užití

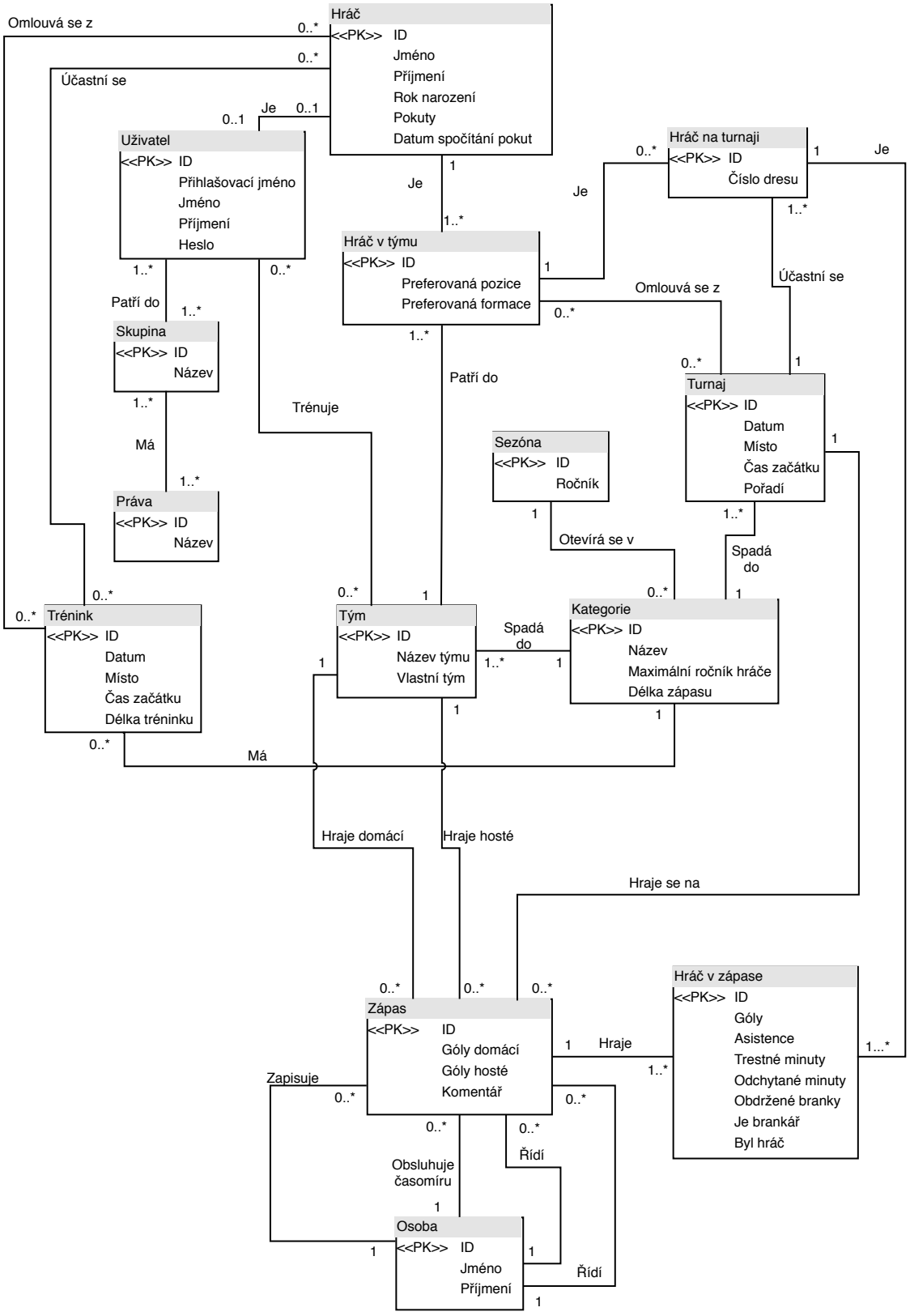
Kapitola 5

Návrh informačního systému

V předchozí kapitole byly popsány všechny požadavky na vytvářený systém. Na základě těchto informací je nutné vytvořit datový návrh informačního systému, kterému se bude tato kapitola věnovat. Pro datový návrh bude použit Entity Relationship Diagram (diagram entit a vztahů).

5.1 ER Diagram

Entity Relationship Diagram je grafický prostředek používaný pro datové modelování. Používá se pro modelování struktury dat v systému a vztahů mezi těmito daty. Data jsou modelována jako entity a jejich atributy. Entita je jednoznačně identifikovatelná věc, o které ukládáme informace (její atributy). Pro databáze je neobvyklé ukládat informace o jedné entitě. Často se tedy v kontextu ER diagramu hovoří o entitních množinách. To jsou množiny entit, které sdílí stejné atributy. Příkladem můžou být zaměstnanci firmy. Zaměstnanci jsou entitní množina, která obsahuje entity jednotlivých zaměstnanců. [7] Diagram vytvářeného systému je na obrázku 5.1.



Obrázek 5.1: Entity–Relationship Diagram

5.1.1 Sezóna a Kategorie

Sezóna představuje ročník soutěže, pro který jsou důležitou informací pouze roky, ve kterých se koná. Pro každou sezónu jsou otevírány kategorie, pro které bude probíhat liga. Název kategorie musí být vždy v rámci sezóny unikátní. Tyto dvě entitní množiny slouží zejména pro možnost ukládání historie jednotlivých turnajů a pro kontrolu dat souvisejících s kategorií.

5.1.2 Tým a Hráč

Týmy vždy náleží do některé z kategorií a v rámci této kategorie musí mít unikátní název. Pro tým je ještě nutné ukládat, zda se jedná o vlastní tým, protože u týmů jiných klubů nebude dostupná pokročilá funkcionalita. Tyto týmy slouží pouze pro vedení statistik turnajů.

U jednotlivých hráčů budou ukládány pouze informace, které se nebudou v průběhu času měnit nebo jsou specifické pro konkrétní hráče napříč všemi ročníky a kategoriemi. Neměnnými informacemi jsou jméno, příjmení a rok narození. Specifickými jsou pokuty a datum zaplacení pokuty. A to z toho důvodu, že je nutné hráčům počítat pokuty nehledě na kategorie a ročníky. Hráč totiž může mít nezaplacené pokuty z minulé sezóny, případně může hrát v jedné sezóně za týmy ve více různých kategoriích.

5.1.3 Hráč v týmu

Hlavní podstatou této entitní množiny je zamezit duplicitním datům, které by vznikaly v případě přímého vztahu mezi hráčem a týmem. Prvním problémem by bylo to, že standardní pozice a formace může být u hráče specifická pro jednotlivé týmy, to by se dalo řešit přepisováním těchto informací při přiřazení hráče do dalšího týmu. To se ale nejevilo jako vhodné řešení a to hlavně z důvodu, že by tento způsob řešení fungoval pouze za situace, kdy by hráč mohl v jedné sezóně být pouze v jednom týmu. Zde vzniká druhý problém: každý hráč, který splňuje věkové podmínky příslušných kategorií, může v jedné sezóně hrát za více týmů v různých kategoriích. V takovém případě by se již duplicitám nedalo vyhnout, a to kvůli již zmiňované preferované pozici a formaci, a také kvůli statistikám jednotlivých hráčů.

Zároveň tato entitní množina slouží i jako soupiska týmu pro danou sezónu. Za tým mohou nastoupit pouze hráči, kteří jsou umístěni na soupisce. Soupiska má i další využití, lze díky ní zjišťovat, kteří hráči patří do vlastních týmů a je tedy nutné jim zpřístupňovat pokročilé funkce.

5.1.4 Trénink

Tréninky jsou vytvářeny pro jednotlivé kategorie a je pro ně důležité ukládat, kdy a kde se konají a jejich délku. Místo konání se bude ukládat pouze jako jeden atribut. Slouží totiž pouze pro informaci hráčům a trenérům, proto není nutné vyžadovat jednotný formát a stačí jej ukládat pouze jako jeden řetězec. Vztahy mezi Hráčem a Tréninkem slouží pro vedení statistiky o účasti na trénincích.

5.1.5 Turnaj

Každý turnaj, stejně jako trénink, je pořádán pro některou z kategorií. O turnaji je nutné ukládat několik informací. Jedná se o datum, místo, čas začátku a pořadí turnaje. Pořadí

turnaje ale nelze použít jako primární klíč entity. Vztahuje se totiž vždy k příslušné kategorii a tím pádem se může opakovat. Místo konání je ukládáno pouze jako jeden řetězec. U některých turnajů stačí uvádět pouze tělocvičnu, u některých je nutné uvést i město, kde se turnaj bude konat. Obdobně jako u tréninku tedy není nutné vyžadovat konkrétní formát místa konání.

5.1.6 Hráč na turnaji a Hráč v zápase

Pro každý turnaj musí být vytvořena soupiska hráčů, kteří mohou nastupovat v zápasech na příslušném turnaji. K tomu slouží entitní množina Hráč na turnaji. Zároveň se u této entity vede informace o čísle hráče. To může mít hráč na každém turnaji jiné. Slouží pouze pro jednodušší hlášení gólů a asistencí zapisovateli, ale následně se již v žádných statistikách nevyužívá. Právě proto se ukládá u této entity. U hráčů vlastních týmů zároveň slouží pro zjišťování, zda se turnaje zúčastnili.

Hráč v zápase slouží k ukládání statistik hráčů v zápasech. Tyto statistiky následně slouží pro zobrazování zápisů a tabulek se statistikami (např. pořadí brankářů a střelců). Tato entita obsahuje ještě dva atributy, konkrétně Je brankář a Byl hráč, které nejsou přímo statistiky, ale slouží k jejich zobrazování.

Atribut Je brankář je využívám při editaci a zobrazování zápisu o utkání. Ze statistik totiž nelze zjistit, zda se hráč účastní zápasu jako hráč v poli nebo brankář. Jak již bylo popsáno v kapitole 4.5, hráč se může stát v průběhu zápasu brankářem a budou u něj tedy uloženy statistiky specifické pro hráče v poli i brankáře. Díky tomuto atributu lze jednoduše určit, kam je hráče potřeba umístit v rámci zápisu o utkání.

Pouhá informace o to, zda hráč je, popřípadě není brankářem, může být v některých případech pro zobrazování zápisu nedostatečná. Pokud se hráč v průběhu zápasu stane brankářem, je nutné ho v zápisech uvádět jako hráče i brankáře. To slouží jednak k informaci, že došlo ke změně pozice daného hráče, a také je nutné zobrazovat jeho statistiky na obou zmíněných postech. K tomu slouží atribut Byl hráč, který je při zmiňované změně nastavován.

5.1.7 Uživatel, Skupina a Práva

Tyto tři entitní množiny slouží ke správě uživatelů a jejich práv. Díky nim bude v systému možné omezovat přístup k jednotlivým funkcionalitám. Název skupiny musí být unikátní a každý uživatel musí spadat alespoň do jedné z nich. Stejně tak i název práva musí být unikátní a každá skupina musí mít alespoň jedno. Toto řešení umožňuje uživatelům efektivně přiřazovat práva, protože je stačí přiřadit pouze do příslušné skupiny, která může mít velké množství práv. Tím se zamezí situaci, kdy by se musela uživatelům přiřazovat práva jednotlivě a vznikalo by velké množství vztahů mezi právem a uživatelem.

V aktuálním systému by bylo možné použít i variantu se dvěma entitními množinami, protože v systému nebude obsaženo velké množství práv a nevznikalo by tedy již zmíněné velké množství vztahů. Ale z hlediska budoucího rozvoje systému je univerzálnější řešení s použitím třech entitních množin.

5.1.8 Zápas a Osoba

O zápase je nutné ukládat relativně velké množství informací. Většina z nich je ve formě jiné entity, která je s entitou Zápas ve vztahu. U samotného zápasu je ukládané pouze skóre domácího a hostujícího týmu a komentář. Ačkoliv jsou vstřelené a obdržené branky

ukládány u entity Hráč v zápase, může nastat situace, kdy z těchto hráčských statistik nelze dopočítat konečné skóre týmů, a proto musí být skóre uloženo přímo u zápasu. Touto situací jsou vlastní góly a góly v powerplay (hře bez brankáře). U hráčů není vedena statistika vlastních gólů. Pokud si tedy některý z hráčů dá vlastní gól, tak není v žádné jeho statistice uložen. Podobně pokud hraje tým bez brankáře a dostane gól, není tento gól uložen v žádné statistice brankáře. Tyto góly by následně chyběly v celkovém skóre týmů.

Veškeré další informace potřebné pro vytváření, editaci a zobrazování zápisu o utkání jsou uloženy v jiných entitách. Toto řešení je zvoleno ze dvou důvodů. Prvním je zamezení ukládání redundantních dat. Druhým důvodem je to, že mnoho dat spojených se zápasem je potřebných i pro jiné entity. Proto by bylo velmi nepraktické je ukládat pouze v jedné entitě.

První z těchto entit je Turnaj. Každý zápas se musí odehrávat na některém z turnajů a je o něm uveden údaj na zápise. Dále se jedná o entitu Tým. Aby se zápas mohl odehrát, musí se ho zúčastnit dva týmy, zároveň je nutné rozlišit domácí a hostující tým. Kromě skóre obou týmů musí být pro každý zápas vedena i osobní statistika hráčů, to zajišťuje entita Hráč v zápase.

Poslední entitou potřebnou pro zápas je Osoba. Na zápise o utkání je vždy uvedeno jméno zapisovatele, časoměřiče a dvou rozhodčích. Tato jména by se často opakovala, a proto by bylo nevhodné je ukládat přímo v entitě Zápas. Jak již bylo zmíněno, pro tyto osoby nejsou vytvářeny uživatelské účty. Vzhledem k tomu bylo nutné vytvořit novou entitu, která bude obsahovat dva atributy potřebné pro osoby (jméno a příjmení).

5.2 Transformace ER diagramu na relační databázi

Všechny entitní množiny se převedou na tabulky relační databáze, kde primární klíče budou stejné jako v ER diagramu. U vztahů 1:1 a 1:N bude vždy u jedné entity přidán ještě cizí klíč, který bude odkazovat na primární klíč druhé z entit. Pro vztahy M:N musí být vytvořena vazební tabulka. V našem případě bude vždy obsahovat dva atributy, které dohromady tvoří primární klíč. Zároveň každý z těchto atributů bude cizím klíčem odkazujícím na jednu z tabulek původního M:N vztahu.

Kapitola 6

Implementace

Na základě výše uvedených požadavků a návrhů je nutné samotný systém implementovat. Tím se bude zabývat následující kapitola, ve které bude popsána implementace nejdůležitějších, popřípadě zajímavých částí systému.

6.1 Databáze

Pro fungování aplikace je nutné vytvoření databáze, kde budou ukládána všechna data. Nejdříve bylo nutné definovat jednotlivé tabulky a vztahy na základě schématu databáze. V SQLAlchemy je toto řešeno pomocí tříd. Pro každou tabulku databáze je tedy vytvořena jedna třída. Jednotlivé proměnné příslušných tříd potom představují sloupce dané tabulky.

Cizí klíče jsou definovány pomocí třídy `ForeignKey`, která se použije jako argument třídy `Column` při vytváření sloupce tabulky. Dále je ve třídách obsahujících cizí klíč vytvořena další proměnná, k jejímu vytvoření je využita metoda `relationship`. Ta slouží k vytvoření vztahu mezi dvěma tabulkami. Díky této proměnné je při načtení dat z databáze možné přistupovat přímo na objekty jiných tabulek, které jsou s načtenou tabulkou spojeny pomocí cizího klíče.

Pro vztahy M:N je nutné vytvořit ještě vazební tabulku. Obdobně jako u ostatních vztahů bude v příslušných třídách pomocí metody `relationship` vytvořena proměnná. Rozdílem ale je to, že pomocí argumentu této metody se proměnná bude odkazovat na zmiňovanou vazební tabulku. Vazební tabulky již nejsou vytvářeny jako třídy. Pro jejich vytvoření je použita třída `Table`, kterou poskytuje SQLAlchemy. Jak již bylo zmíněno, vazební tabulky obsahují dva sloupce, které se odkazují na primární klíče původních tabulek a zároveň spolu tvoří primární klíč vazební tabulky.

6.2 Vytváření uživatelů a přihlašování

Vytváření uživatelů je možné dvěma způsoby. Jedním z nich je automatické vytváření uživatelských účtů. To se využívá výhradně pro vytváření účtů pro hráče. Pro všechny ostatní je již možné vytvářet účty klasickým způsobem, kdy je zadáno jméno, příjmení, uživatelské jméno a heslo.

Při automatickém vytváření účtu je použito jméno a příjmení hráče, pro kterého je účet vytvářen. Uživatelské jméno je tvořeno příjmením hráče, dvěma prvními písmenky jména hráče a číslem od 0 do 99, které je náhodně generováno. Pokud by se tedy stalo, že bude mít více hráčů stejné jméno a příjmení, budou jejich uživatelská jména odlišena právě tímto

číslem. Heslo je také generováno automaticky a skládá se z náhodných velkých a malých písmen a číslic.

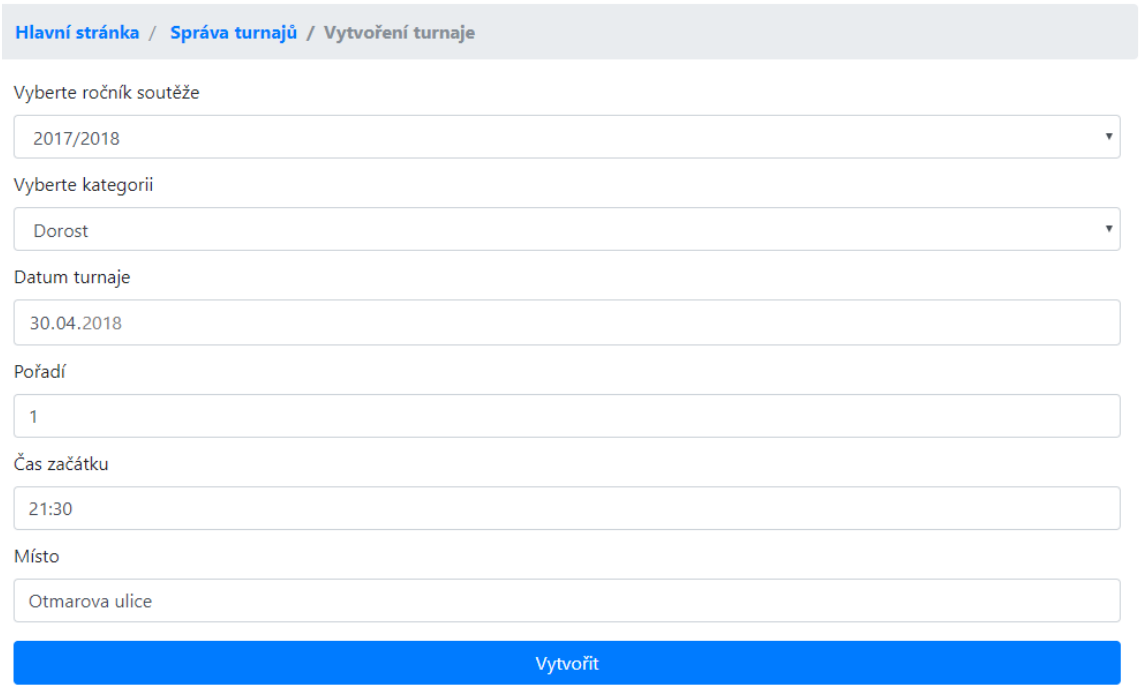
Z bezpečnostních důvodů je nevhodné ukládat hesla v databázi jako text. Hesla jsou tedy v databázi ukládána jako hash, k tomu je využita funkce `generate_password_hash`, která používá algoritmus PBKDF2 s SHA512 hashem. Pokud by byla hesla z databáze odcizena, není možné je použít přímo, ale bylo by nutné nejdříve prolomit šifrování.

Při přihlašování uživatel zadává uživatelské jméno a heslo. Pomocí uživatelského jména je nalezen příslušný uživatel v databázi. Z databáze je získáno jeho heslo, které je pomocí funkce `check_password_hash` porovnáno se zadaným heslem. Pokud je uživatel nalezen a zadané heslo se shoduje s tím z databáze, je uložen objekt přihlášeného uživatele a zároveň jsou načtena všechna jeho práva, která jsou následně kontrolována při přístupech na jednotlivé stránky.

6.3 Formuláře

Formuláře jsou stěžejní pro celý informační systém. Používají se pro vytváření veškerých dat, která jsou pro fungování systému nutná. Zároveň je nutné ukládaná data zobrazovat, a vzhledem k tomu, že by bylo nepraktické zobrazovat například statistiky ligy pro všechny turnaje a ročníky zároveň, jsou formuláře využívány i k upřesnění dat, která chce uživatel zobrazit.

Vzhled formulářů je vytvářen s pomocí knihovny Bootstrap. Pro většinu formulářů je typické, že jsou postupně vybírány hodnoty (např. vybrání ročníku, následně kategorie a týmu v této kategorii). Proto jsou jednotlivá pole umístěna pod sebou. Pro konzistentnost vzhledu jsou tímto způsobem umístěny i formulářové prvky, kde je nutné zadávat nějaký text, případně číslo. Díky použití knihovny Bootstrap nemají jednotlivé prvky pouze základní vzhled. Navíc jsou všechny formuláře responzivní. Příklad formuláře je na obrázku 6.1.



Obrázek 6.1: Příklad formuláře

Při zobrazování formulářů je nutné dynamicky upravovat možnosti pro výběr na základě předchozí volby uživatele. Data pro tyto možnosti jsou získávána ze serveru. Jak již bylo zmíněno v kapitole 3.5, pro tyto účely se skvěle hodí využít AJAX. Při výběru hodnoty uživatelem je na server vyslán asynchronní požadavek, jehož součástí jsou i vybrané hodnoty. Na serveru je následně požadavek zpracován a získaná data jsou odeslána zpět klientovi. Tam jsou pomocí JavaScriptu zpracována a přidána k příslušnému HTML elementu. Díky tomuto postupu není nutné znovu načítat stránku a pro uživatele je práce s formuláři plynulejší. Stejný postup je používán i u formulářů, kde jsou načítáni a zobrazováni například hráči nebo tréninky.

Pro zpracování zadávaných dat je nutná i validace formulářů. Ta je zajištěna pomocí JavaScriptu přímo u klienta. Pokud by totiž validace probíhala až na serveru, mohlo by se stát, že uživatel zapomene vyplnit jedno formulářové pole a stránku by bylo nutné po odeslání formuláře znovu načítat. Teprve v tuto chvíli by uživatel zjistil, že formulář nevyplnil správně a bylo by nutné celý postup opakovat. Při validaci na straně klienta je odeslání formuláře zabraňováno, dokud není vyplněn správně. Zároveň je při pokusu o odeslání uživateli vypsána zpráva, které ho upozorňuje na konkrétní údaje, které je nutné ještě doplnit. Při odstraňování hráčů a tréninků je navíc uživatel dotázán, zda si je opravdu jistý, že chce tuto operaci provést. Tím se zabrání nechtěnému odstranění dat. Zároveň po úspěšné validaci již nelze znovu použít tlačítko pro odeslání formuláře, aby nedocházelo k vícenásobnému odeslání dat.

Pokud byl formulář odeslán a uživatel by se chtěl vrátit zpět na předchozí stránku nebo obnovit danou stránku, je prohlížečem tázán, zda chce znovu odeslat formulář. Toto chování není příjemné pro uživatele a navíc může způsobit nechtěné znovuodeslání dat. Toto

je řešeno tak, že po odeslání formuláře je uživateli zobrazena jiná stránka, kde je uvedena informace o stavu provedené operace. Zároveň je možné se vrátit zpět na formulář nebo obnovit stránku, aniž by došlo k opětovnému odeslání dat.

6.4 Zobrazování sestavy

Pro zobrazování sestavy jsou využívány informace o standardních formacích a pozicích hráčů. Ale dle těchto informací by bylo možné zobrazovat pouze preferovanou sestavu. Ve většině případů se ale někdo z hráčů omluví, popřípadě se na turnaj nedostaví. Pro takové případy je nutné mít stanovený způsob náhrady těchto hráčů.

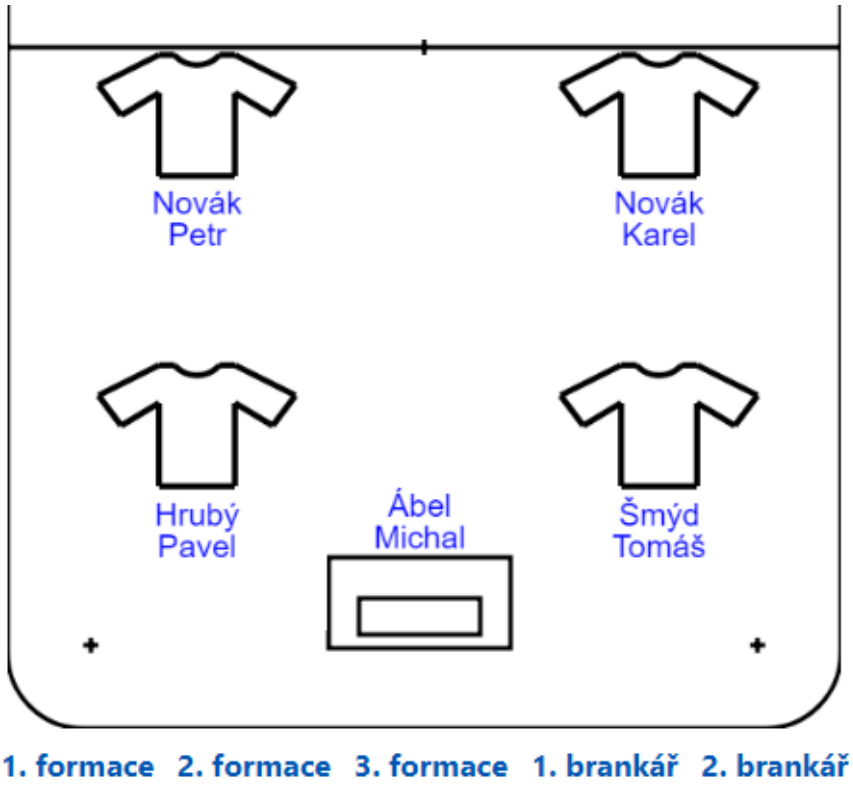
Nejjednodušší situace nastává ve chvíli, kdy v dané formaci a na příslušné pozici chybí pouze jeden hráč. Ten je nahrazen hráčem ze stejné pozice. Pokud v žádné jiné formaci není hráč na stejné pozici, musí ho nahradit hráč z jiné pozice. Pokud není k dispozici ani takový hráč, nelze chybějícího hráče nahradit a formace musí zůstat s nižším počtem hráčů. Pokud je tím jedním chybějícím hráčem brankář, je nahrazen jedním z hráčů poslední formace. V systému totiž není ukládána informace o tom, který z hráčů by v případě potřeby mohl nastoupit jako brankář. Jedná se totiž o velmi specifickou situaci, která nastává velmi zřídka. Její řešení pak záleží na domluvě mezi hráči a trenéry. Může být rozhodnuto i na základě toho, proti komu se hraje. Vzhledem k těmto okolnostem bylo dohodnuto, že při vykreslování sestavy bude vždy vybrán jeden z hráčů, aniž by bylo nutné ho vybírat podle nějakých kritérií.

Složitější situace nastává, pokud na dané pozici chybí ve formaci dva hráči. V takové situaci je nejlepší možností nahradit je dvěma hráči z jiné formace, ale stejné pozice. Tito hráči spolu běžně hrají, jsou na sebe tedy zvyklí a je to lepší varianta, než k sobě dávat hráče z jiných formací. Pokud nejsou takovýto hráči nalezeni, je nutné přistoupit k jiným možnostem. První možností je přidání dvou hráčů, kteří hrají na stejné pozici, ale jsou z různých formací. Tito hráči spolu sice běžně nehrají, ale pořád alespoň hrají na stejné pozici jako chybějící hráči. V případě, že je nalezen pouze jeden hráč ze stejné pozice, je nutné druhého hráče nahradit hráčem z jiné pozice. Pokud by se nenašli žádní hráči ze stejné pozice, musí místo nich hrát hráči z jiné pozice. V tomto případě probíhá vyhledávání dostupných hráčů obdobně jako v předchozím případě. Nejdříve jsou vyhledávání hráči z jedné formace. I když obvykle hrají na jiné pozici, je alespoň výhodou, že běžně hrají spolu. Pokud nejsou takoví hráči nalezeni, existuje již pouze poslední možnost jak je nahradit. Jsou tedy vyhledáni hráči z jiné pozice a různých formací. Toto je již krajní případ, kdy neexistuje žádná dvojice hráčů, kteří by spolu běžně hráli.

Při vyhledávání dostupných hráčů se vždy začíná od formací s nejnižším číslem. V extrémních případech, kdy chybí hodně hráčů, to může způsobit větší vytěžování některých hráčů, zatímco jiní nebudou v tolika formacích. Takovéto situace již bohužel není možné řešit automaticky. Vždy to bude záležet na fyzické připravenosti daných hráčů, domluvě s trenérem, popřípadě je možné střídat formace v takovém pořadí, aby tito hráči nehráli bezprostředně za sebou.

Pro samotné vykreslení sestavy je použit JavaScript a HTML element `Canvas`. Nejprve je vždy vykreslen plánek hřiště. Následně je vždy automaticky zobrazena jedna formace. Pro každého hráče z této formace je vykreslen dres a pod něj je napsáno jeho jméno. Pouze u brankáře není vykreslen dres. Místo toho je jeho jméno zobrazeno u brankoviště. V případě, že uživatel chce zobrazit jinou formaci, jsou smazány všechny dresy a jména hráčů aktuální formace. Následně je znovu opakováno vykreslení dresů a jmen. Smazání dresů probíhá kvůli tomu, že následující formace může mít jiný počet hráčů (například

pokud se omluvilo velké množství hráčů). Kdyby nebyly smazány dresy, ale pouze jména, zůstaly by pak při vykreslení další formace některé dresy bez jména. To by mohlo být pro uživatele matoucí. Příklad zobrazené sestavy je na obrázku 6.2.



Obrázek 6.2: Příklad grafického zobrazení sestavy

6.5 Zápis zápasu

Při vytváření zápisu zápasu je nutné řešit několik zajímavých situací, které byly nastíněny v kapitole 4.5. Dále je také vhodné ukládat všechny statistiky průběžně, aby se při případném výpadku připojení podařilo uložit co nejvíce dat. Pokud by se statistiky ukládaly naráz po ukončení zápasu, mohlo by se stát, že se je nepodaří uložit a bylo by nutné vše zadávat znovu. U zápisu je nutné ukládat ještě poznámky, ty jsou jako jediné ukládány až při odeslání formuláře.

Veškeré zadávané statistiky jsou ukládány ihned po změně, k tomu je využíváno AJAXu. Při změně hodnoty některé ze statistik je odeslán asynchronní požadavek na server, kde je zpracován. To znamená uložení změněné statistiky, případně všech statistik, které jsou změněnou hodnotou ovlivněny. Následně je nutné připravit hodnoty všech změněných statistik a odeslat je zpět klientovi na zpracování. Součástí odesílaných dat i je indikátor úspěchu operace.

Po přijetí dat klientem je nutné provést několik operací v závislosti na indikátoru úspěchu. V případě neúspěchu se jedná pouze o zapsání resetované hodnoty, která přišla ze serveru. Tato hodnota se zapíše do vstupu pro statistiku, která byla původně změněna. Zároveň je nutné vypsat uživateli informaci o neúspěšném uložení statistiky. V případě

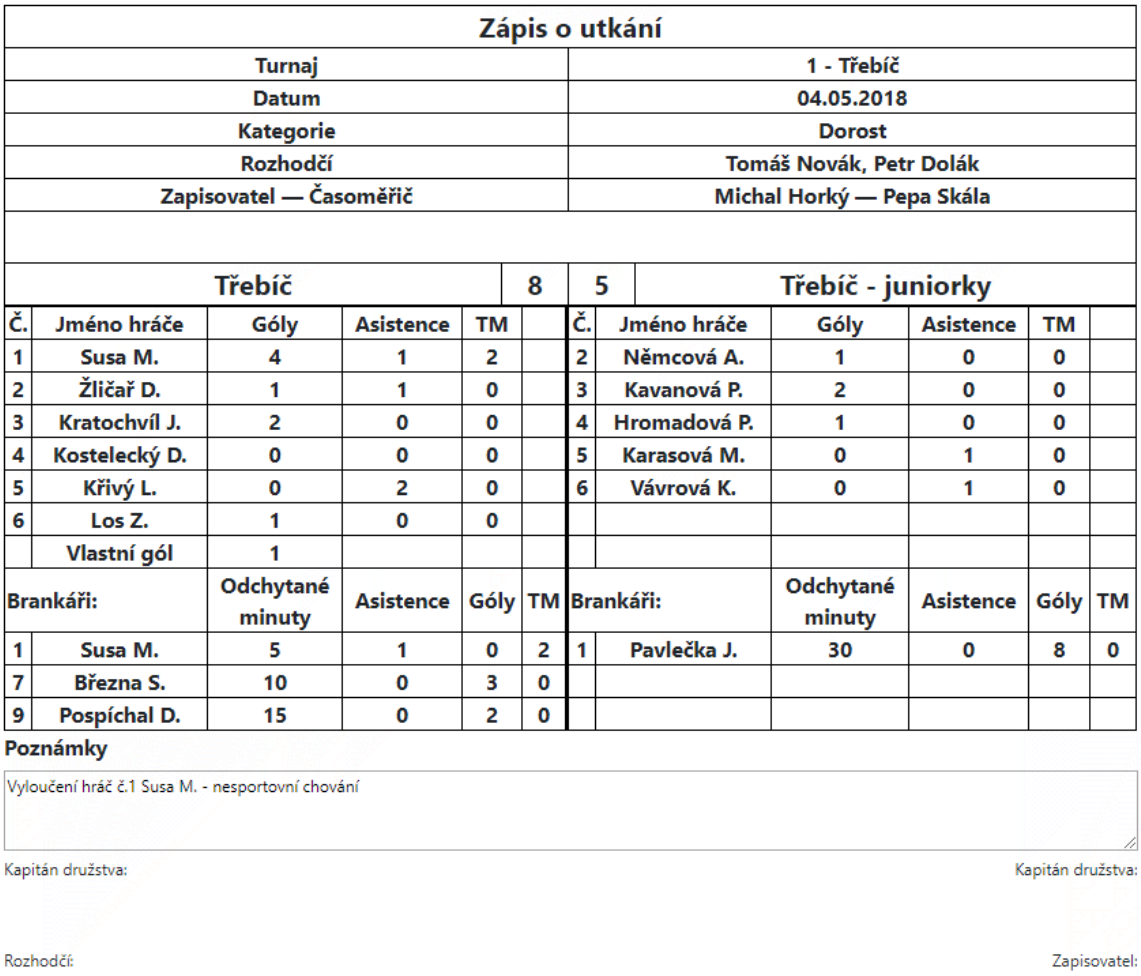
úspěchu je několik možností, které mohou následovat. Na některých statistikách nejsou závislé žádné další. Těmito statistikami jsou asistence, trestné minuty a odchytané minuty. Při jejich změně není při úspěšné operaci nutné měnit nic dalšího. Při změně vstřelených gólů je již nutné změnit i závislé statistiky. Pokud je některému z hráčů změněn počet vstřelených gólů, je nutné změnit počet gólů týmu a zároveň změnit počet obdržených branek právě chytajícímu brankáři.

Jednou ze zajímavých situací jsou vlastní góly a góly v powerplay. Pro oba případy je vytvořen speciální řádek v zápise. Při vlastním gólu je stejně jako u gólu vstřeleného hráčem nutné změnit celkový počet gólů týmu a počet obdržených branek brankáře. Rozdílem je to, že gól není připsán žádnému hráči a do obdržených branek je připsán vlastnímu brankáři, nikoliv brankáři soupeře. Zpětně je počet vlastních gólů dopočítáván jako rozdíl počtu obdržených branek týmu a součtu vstřelených gólů všech hráčů soupeře. U gólů v powerplay je naopak nutné obdržené branky nepřipisovat žádnému z brankářů. Vstřelený gól je tedy připsán příslušnému hráči a týmu, ale již není nikde ukládán jako obdržený. Góly obdržené v powerplay je možné dopočítat jako rozdíl obdržených branek týmu a součtu obdržených branek všech brankářů příslušného týmu. Pokud si dá některý z hráčů vlastní gól při hře bez brankáře, je tento gól připočítán pouze ke gólům týmu, ale není již ukládán u žádného z hráčů.

Další situací je střídání brankářů, případně změna hráče v poli na brankáře. Pro střídání brankářů slouží `radio button`, který je u jména každého brankáře. Aktivní `radio button` značí aktuálně chytajícího brankáře. Při změně brankáře je automaticky dopočítán čas, po který se předpokládá, že bude chytat. Ten se získá jako rozdíl délky zápasu a času, po který chytali ostatní brankáři. Pro změnu hráče z pole na brankáře slouží tlačítko, které je u jména každého hráče. Při změně jsou u hráče nastaveny příslušné příznaky, které následně slouží pro rozhodování, kam hráče umístit v zápise. Po této změně je znovu načtena stránka, aby byl příslušný hráč umístěn mezi brankáře. Nadále zůstává zapsán i mezi hráči, pokud totiž vstřelil branku, vznikl by rozdíl mezi zobrazenými góly týmu a vstřelenými góly jednotlivých hráčů, což by mohlo zapisovatele zmást. Jelikož je pak hráč veden na dvou místech, je nutné synchronizovat asistence a trestné minuty, které už je ale možné měnit pouze v řádku brankáře.

Statistiku obdržených gólů týmu a obdržených gólů brankářů jsou vždy změněny pouze na základě vstřelených gólů. Tím se zamezí tomu, že zapisovatel omylem uvede jiný počet vstřelených a obdržených branek. To by mohlo způsobit chyby ve statistikách nebo by to vyžadovalo kontrolovat počty gólů před odesláním formuláře. Proto byl zvolen popisovaný přístup, který se jevil jako nejjednodušší a zároveň nejbezpečnější.

Pro zobrazování zápisu je vytvořena další stránka. To umožňuje ze zápisu odstranit nepotřebné prvky, jako jsou tlačítka pro změnu hráče na brankáře, prvky pro výběr aktivního brankáře či vstup pro statistiky. Ty je možné zobrazovat pouze jako text. Zároveň je ještě přidán prostor pro podpisy kapitánů, rozhodčích a zapisovatele, protože je nutné zápis po každém zápase vytisknout a nechat podepsat všemi zmíněnými. Zároveň umístění zápisu pro zobrazení na jinou stránku umožňuje efektivnější kontrolu uživatelských práv. Zobrazit zápis může i nepřihlášený uživatel, zatímco upravovat zápis může pouze zapisovatel nebo organizátor. Příklad zápisu je na obrázku 6.3.



Zápis o utkání

Turnaj	1 - Třebíč
Datum	04.05.2018
Kategorie	Dorost
Rozhodčí	Tomáš Novák, Petr Dolák
Zapisovatel — Časoměřič	Michal Horký — Pepa Skála

Třebíč						8	5	Třebíč - juniorky					
Č.	Jméno hráče	Góly	Asistence	TM			Č.	Jméno hráče	Góly	Asistence	TM	
1	Susa M.	4	1	2			2	Němcová A.	1	0	0	
2	Žličař D.	1	1	0			3	Kavanová P.	2	0	0	
3	Kratochvíl J.	2	0	0			4	Hromadová P.	1	0	0	
4	Kostelecký D.	0	0	0			5	Karasová M.	0	1	0	
5	Křivý L.	0	2	0			6	Vávrová K.	0	1	0	
6	Los Z.	1	0	0								
	Vlastní gól	1										
Brankáři:		Odchytané minuty	Asistence	Góly	TM		Brankáři:		Odchytané minuty	Asistence	Góly	TM
1	Susa M.	5	1	0	2		1	Pavlečka J.	30	0	8	0
7	Března S.	10	0	3	0							
9	Pospíchal D.	15	0	2	0							

Poznámky

Vyloučení hráč č.1 Susa M. - nesportovní chování

Kapitán družstva: Kapitán družstva:

Rozhodčí: Zapisovatel:

Obrázek 6.3: Příklad zobrazeného zápisu o utkání

6.6 Statistiky

V systému jsou uživatelům přístupné různé statistiky, týkající se ligy a účasti hráčů. U některých statistik, kde je důležité pořadí účastníků, jsou kritéria jednoduchá. U jiných naopak může být kritérií několik a vytváření pořadí u těchto statistik je zajímavější. U všech statistik je nutné se nejdříve pokusit o pořadí rozhodnout na základě kritérií pro danou kategorii statistik. Teprve až po vyčerpání všech možností je možné uvádět účastníky na stejném místě. V tomto případě je nutné adekvátně přizpůsobit číslování pozic v příslušné tabulce. Číslo další pozice za tou, na které je více účastníků, musí být posunuto o počet účastníků, kteří jsou na předchozí pozici navíc. Příklad takové tabulky je obrázku 6.4.

Um.	Jméno	Tým	Počet gólů
1.	Martin Susa	Třebíč	6
2.	Petr Vala	Dalešice	4
3.	Anna Němcová	Třebíč - juniorky	3
4.	Josef Diviš	Dalešice	2
	David Kostelecký	Třebíč	2
	Oldřich Bartošík	Třebíč	2
	Matěj Dokulil	Třebíč	2
	Pavlína Kučerová	Třebíč - juniorky	2
9.	Robin Všetečka	Třebíč	1

Obrázek 6.4: Příklad tabulky pořadí hráčů

Pro vyhodnocení jsou nejjednoduššími statistikami trestné minuty, vstřelené góly a kanadské bodování. U vstřelených gólů je jediným možným kritériem počet vstřelených branek, pokud jich má více hráčů stejný počet, jsou ve statistikách uváděni na společném místě. U kanadského bodování se ke gólům připočítají ještě asistence. Zároveň je při vytváření pořadí v případě shodného počtu bodů nutné upřednostnit počet vstřelených branek. U trestných minut není pořadí hráčů ani uváděno, pouze jsou v tabulce seřazeni od nejvyššího počtu trestných minut po nejnižší. Trestné minuty jsou navíc sčítány pro celé týmy. Statistika trestných minut pro týmy je pak zobrazována ve formě koláčového grafu. Mezi zobrazením trestných minut hráčů a týmů je možné se přepínat.

U brankářů je prvním kritériem počet obdržených branek na minutu. Při zobrazování statistik je tento počet uváděn na tři desetinná místa, v rámci systému je ale počítáno s vyšší přesností. U počtu obdržených branek na minutu platí to, že čím nižší toto číslo brankář má, tím výš se v rámci statistiky umístí. Pro pořadí brankářů již další kritérium neexistuje. Výjimku tvoří situace, kdy může být brankář z pořadí vyřazen úplně. Pro brankáře platí pravidlo, že pro zařazení do pořadí brankářů musí odchytat alespoň 40% času. Je tedy nutné spočítat všechny zápasy daného týmu a podle délky zápasu pro příslušnou kategorii vypočítat celkový možný čas, který mohl brankář odchytat. Dle toho je potom možné zjistit, zda brankář splnil podmínku zařazení do statistiky a bude tedy uveden v pořadí brankářů. Pokud podmínku nesplnil, je pro něj vypočítán počet gólů na minutu, ale již není nutné porovnávat tuto hodnotu s ostatními. Bude totiž v tabulce uveden na posledních místech bez udaného pořadí.

Nejzajímavější je vytváření pořadí týmů. Zde je prvním kritériem počet získaných bodů, za výhru tým získává 3 body, za remízu 1 bod a za prohru 0 bodů. Pro každý tým je podle vstřelených a obdržených gólů v jednotlivých zápasech zjištěn celkový počet vítězství, remíz a proher. Na jejich základě je potom vypočítán celkový počet bodů týmu. Při shodném počtu bodů je nutné zkusit vytvořit pořadí týmů na základě dalších kritérií. Tím prvním jsou vzájemné zápasy. Pokud se jedná o dva týmy, je postup jednoduchý. Na vyšší příčce bude vždy tým, který ve vzájemných zápasech získal víc vítězství. Pokud mají oba týmy stejný počet vítězství, je další statistikou, která může rozhodnout o pořadí, skóre. Skóre se získá jako rozdíl vstřelených a obdržených branek týmu (v tomto případě ale pouze ze

vzájemných zápasů s týmem, se kterým mají stejný počet bodů). Může se ale stát, že stejný počet bodů budou mít více než dva týmy. V takové případě již není možné brát v potaz pouze vyšší počet vítězství, ale je nutné vytvořit minitabulku daných týmů. Do této tabulky jsou započítávány pouze body ze vzájemných zápasů těchto týmů. I v této minitabulce může vzniknout shoda bodů. V tomto případě se již nevytváří další minitabulka týmů se stejným počtem bodů, ale rozhoduje skóre těchto týmů ze vzájemných zápasů s týmy z minitabulky. Pokud nebude možné rozhodnout o pořadí týmů ani podle tohoto kritéria, zůstává pouze poslední statistika, dle které je možné určit pořadí. Touto statistikou je počet vstřelených gólů příslušných týmů ve vzájemných zápasech.

Kapitola 7

Testování

Nezbytnou součástí vývoje systému je jeho testování. To slouží k odhalení chyb a nedostatků, které je nutné před uvedením systému do provozu odstranit. Zároveň slouží k prokázání, že systém obsahuje požadovanou funkcionalitu a může pomáhat k ujasnění dalších požadavků na vyvíjený systém. Testování je možné rozdělit do několika kategorií, jedná se například o manuální a automatické nebo dynamické a statické testování.

Veškeré testování vyvíjeného informačního systému probíhalo manuálně. Což znamená, že testování neprobíhalo za pomoci softwaru, ale prováděli ho samotní uživatelé systému, popřípadě programátor. Zároveň se jednalo o dynamické testování, protože k němu bylo třeba běhu aplikace. Pro testování byly dále použity výsledky minulých ročníků ligy, s jejich pomocí bylo možné testovat statistiky turnajů.

7.1 Testování programátorem

Samotný programátor by měl vyvíjený systém testovat neustále. Výhodou je i to, že z časových důvodů nemusí být možné, aby uživatelé testovali jednotlivé přidávané funkcionality ihned. Kvůli tomu by mohly při vývoji vznikat nepříjemné časové odstupy před přidáváním dalších funkcionalit. Přesně tato situace nastala i při vývoji tohoto informačního systému. Programátor testoval každou nově přidanou funkcionalitu. Uživatelské testování bylo prováděno až ve chvíli, kdy byla vytvořena skupina funkcí, které tvořily některou část z požadováného systému. Zároveň již tyto funkce byly otestovány programátorem, který již u daných funkcí nenacházel další chyby.

7.2 Uživatelské testování

Toto testování probíhalo vždy po vytvoření některé části systému. Ačkoliv se předpokládá, že systém budou používat výhradně členové florbalového klubu, pro uživatelské testování byli přizvání i lidé, kteří nejsou členy klubu, nejsou seznámeni s tímto sportem a správou klubu. I přesto, že byly jednotlivé funkcionality otestovány programátorem, dalo se předpokládat, že se mohou objevit i další chyby. Ty mohou být způsobeny například odlišným způsobem používání systému jednotlivými uživateli. Zároveň bylo možné díky tomuto testování získat i další důležité podněty, které jsou popsány v následujících odstavcích.

U členů klubu se předpokládalo, že pro ně bude jednodušší provádět jednotlivé akce v systému. A to především díky znalosti jednotlivých souvislostí potřebných pro organizaci všech akcí pořádaných klubem (tréninků, turnajů a ligy) a všech informací, které jsou k této

organizaci potřeba. Tento předpoklad se potvrdil a díky tomu byly při testování těmito uživateli rychleji odhalovány chyby funkcionality. Zároveň měli uživatelé možnost vyjádřit se k ostatním aspektům systému jako jsou například informace zobrazované u formulářů, rozmístění některých prvků na stránce nebo formát samotných formulářů.

Při testování uživateli, kteří nejsou členy klubu, bylo hlavním cílem získání zpětné vazby ohledně přehlednosti uživatelského rozhraní. Vzhledem k tomu, že tito uživatelé nejsou seznámeni s náležitostmi potřebnými pro pořádání jednotlivých akcí klubu, bylo pro tyto uživatele mnohem důležitější, aby byl systém přehledný a naváděl je k vytvoření všech potřebných náležitostí pro pořádání příslušné akce. Při tomto testování byla získávána především zpětná vazba ohledně informací zobrazovaných u formulářů, rozmístění některých prvků apod. I při tomto testování bylo samozřejmě možné odhalovat chyby funkcionality.

Posledními uživateli, kteří systém testovali, byli samotní zadavatelé. Jednalo se tedy především o pořadatele ligy a trenéry. Při tomto testování šlo především o zpětnou vazbu ohledně samotných funkcionalit. Tito uživatelé tedy testovali, zda systém obsahuje všechny požadované funkcionality, případně zda je možné v systému ukládat všechny potřebné informace. Hlavní výhodou tohoto testování bylo doplňování požadavků, na které se z nějakého důvodu zapomnělo. Zároveň bylo možné diskutovat samotnou funkcionalitu a upravovat ji dle potřeb zadavatelů.

Kapitola 8

Závěr

Cílem této práce bylo vytvořit informační systém pro organizaci akcí pořádaných florbalovým klubem. Zadavatelem byl florbalový oddíl Tělocvičné jednoty Sokol Třebíč, pro který již bylo neúnosné řešit veškerou organizaci pořádaných akcí pouze za pomoci zápisů v textových souborech.

Na začátku bylo nutné provést analýzu požadavků, která vymezuje funkcionalitu celého systému. Bez ní by tedy nebylo možné systém vyvíjet. Analýza byla úspěšně provedena díky konzultacím se zadavatelem. Zároveň byla postupně doplňována o některé drobnosti na průběžných schůzkách se zadavatelem. Výsledkem analýzy požadavků byl diagram případů užití, který graficky znázorňuje funkcionalitu systému a zároveň je z něj patrné, jaké možnosti mají jednotliví uživatelé.

Na základě analýzy požadavků bylo možné vytvořit návrh informačního systému. Jeho stěžejní částí bylo vytvoření ER diagramu, který zobrazuje strukturu dat v systému a vztahy mezi těmito daty. Zároveň bylo nutné vybrat technologie, které budou pro implementaci použity. Nakonec bylo rozhodnuto, že systém bude implementován v jazyce Python a jeho frameworku Flask.

Systém byl průběžně testován autorem, členy klubu a lidmi mimo klub. Z testování vyplynulo několik zajímavých podnětů, které byly postupně implementovány. Zároveň byly na základě testování opraveny všechny odhalené chyby. Výsledkem je informační systém, který splňuje všechny požadavky zadavatele. Vzhledem k tomu, že v době dokončení práce byl již ukončen aktuální ročník, je jeho nasazení plánováno na začátek nové sezóny, tedy na září roku 2018. Zároveň bude systém dále vyvíjen dle podnětů, které mohou vzniknout při ostrém používání systému.

Zajímavou možností dalšího vývoje systému by bylo umožnit systém používat i ostatním týmům z ligy. V takovém případě by se již nejednalo o informační systém florbalového klubu, ale spíše florbalové ligy. Toto rozšíření by mohlo usnadnit práci organizátorům. Například by bylo možné, aby si soupisku týmu pro turnaj vytvářely samotné týmy. U těchto soupisek by byl vždy termín, po kterém by již nebylo možné ji měnit. Organizátoři by již tedy nemuseli soupisky všech týmů zadávat sami těsně před začátkem turnaje. Zároveň by bylo možné zobrazovat veškeré statistiky pouze v rámci týmů, díky tomu by hráči získali snadnější přístup ke statistikám svých spoluhráčů. Takový systém by již měl potenciál pro použití i v ligách, které probíhají v jiných krajích, což by zároveň umožňovalo všem příznivcům ligy jednoduše získávat informace o jiných soutěžích, o tom, kdo má šanci postoupit do oblastních přeborů apod.

Literatura

[1] *HTML Introduction*. [Online; navštíveno 05.04.2018].
URL https://www.w3schools.com/html/html_intro.asp

[2] *HTML Styles - CSS*. [Online; navštíveno 05.04.2018].
URL https://www.w3schools.com/html/html_css.asp

[3] *Informace o webových aplikacích*. [Online; navštíveno 31.03.2018].
URL https://helpx.adobe.com/cz/dreamweaver/using/web-applications.html

[4] *Template Engines*. [Online; navštíveno 05.04.2018].
URL https://www.fullstackpython.com/template-engines.html

[5] *Welcome to Jinja2*. [Online; navštíveno 05.04.2018].
URL http://jinja.pocoo.org/docs/2.10/

[6] *What is PHP?* [Online; navštíveno 05.04.2018].
URL http://php.net/manual/en/intro-whatis.php

[7] Bagui, S.; Earp, R.: *Database Design Using Entity–Relationship Diagrams*. CRC Press LLC, 2003, ISBN 978-0-8493-1548-0.

[8] Copeland, R.: *Essential SQLAlchemy*. O'Reilly Media, Inc., 2008, ISBN 978-0-596-51614-7.

[9] Daoust, N.: *UML requirements modeling for business analysts*. Technics Publications, LLC, 2012, ISBN 978-1-9355042-4-5.

[10] Fitzgerald, M.: *Learning Ruby*. O'Reilly Media, Inc., 2007, ISBN 978-0-596-52986-4.

[11] Harms, D.; McDonald, K.: *Začínáme programovat v jazykce Python*. Computer Press, a.s., 2008, ISBN 978-80-251-2161-0.

[12] Holzner, S.: *AJAX: A Beginner's Guide*. The McGraw-Hill Companies, 2008, ISBN 978-0-07-149429-8.

[13] Python, R.: *Model-View-Controller (MVC) Explained – With Legos*. [Online; navštíveno 11.04.2018].
URL https://realpython.com/the-model-view-controller-mvc-paradigm-summarized-with-legos/

[14] Ronacher, A.: *Foreword*. [Online; navštíveno 05.04.2018].
URL http://flask.pocoo.org/docs/0.12/foreword/

[15] Wagner, B.: *Introduction to the C# Language and the .NET Framework*. [Online; navštíveno 05.04.2018].
URL https://docs.microsoft.com/cs-cz/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework

[16] Čápka, D.: *Popis MVC architektury*. [Online; navštíveno 06.04.2018].
URL https://www.itnetwork.cz/php/mvc/objektovy-mvc-redakcni-system-v-php-popis-architektury

[17] Žára, O.: *JavaScript: programátorské techniky a webové technologie*. Computer Press, 2015, ISBN 978-80-251-4573-9.

Příloha A

Obsah CD

- **/xhorky23/src/** – zdrojové kódy informačního systému
- **/xhorky23/tex/** – zdrojový tvar písemné zprávy
- **/xhorky23/BP/** – písemná zpráva ve formátu pdf
- **/xhorky23/instalace/** – návod k instalaci ve formátu pdf